5

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/651,381	08/29/2003	Kalpesh D. Mehta	110349-133005	9259

25943 7590 06/25/2007

SCHWABE, WILLIAMSON & WYATT, P.C.
PACWEST CENTER, SUITE 1900
1211 SW FIFTH AVENUE
PORTLAND, OR 97204

EXAMINER	
HOLLOWAY, DAVID A	

ART UNIT	PAPER NUMBER
2109	

MAIL DATE	DELIVERY MODE
06/25/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/651,381	MEHTA, KALPESH D.	
	Examiner	Art Unit	
	David A. Holloway	2109	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *29 August 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-36* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-36* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *02 September 2003* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *11032003*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-36 are pending.

Drawings

2. The drawings are objected to because Fig. 4c has a spelling error for item #424, "update current threaid". Corrected drawing sheets in compliance with 37 CFR 1.121(d) are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. The figure or figure number of an amended drawing should not be labeled as "amended." If a drawing figure is to be canceled, the appropriate figure must be removed from the replacement sheet, and where necessary, the remaining figures must be renumbered and appropriate changes made to the brief description of the several views of the drawings for consistency. Additional replacement sheets may be necessary to show the renumbering of the remaining figures. Each drawing sheet submitted after the filing date of an application must be labeled in the top margin as either "Replacement Sheet" or "New Sheet" pursuant to 37 CFR 1.121(d). If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

Specification

3. The disclosure is objected to because of the following informalities: the copending application number on p. 11 needs to be filled in. Appropriate correction is required.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1, 2, 6, and 10-12 are rejected under 35 U.S.C.103(a) as being unpatentable over Borkenhagen et al. (US 6,567,839), hereinafter Borkenhagen in view of Torii (US 5,913,059).

6. As to claim 1, Borkenhagen discloses a processing block comprising:
a storage sub-block (Figure 3, #140, main memory, #130, L2 cache, #120, L1 D-cache);
an execution sub-block to execute instructions (Figure 3, #260, #270, and #280); and
a thread management sub-block coupled to the storage and execution sub-blocks (Figure 4A, #450, the thread switch controller is connected to the

execution units through the sequencers and the thread switch controller is connected to the storage sub-block through the sequencers),
and equipped to store and maintain a thread switching
structure in the storage sub-block to facilitate interleaving execution of a plurality of threads of instructions by the execution sub-block (col. 13, lines 33-67, col. 14, lines 1-2, the thread structure is the collection of bits in the thread state register), the thread structure including a current thread identifier identifying one of the plurality of threads as a current thread being currently executed by the execution sub-block (col. 6, lines 14-19), and a thread array (Figure 4A, #440, the state registers comprise the thread array) of thread entries, one per thread (Figure 4A, #440, there is one thread state register per thread), correspondingly describing the plurality of threads, each thread entry being created and added to the thread array (Figure 4A, #400, the state registers comprise the thread array) by the thread management sub-block (col. 12, lines 41-45, Figure 4A and 4B, the instruction unit and the storage control unit are connected to the thread switch logic and data bits from these units update the thread state registers with the new state of a spawned thread, i.e., create thread entries and add them to the thread array).

7. Borkenhagen does not disclose a create thread instruction of a thread that spawns execution of another thread. However, Torii discloses a create thread instructions (col. 4, lines 65-67, child threads can spawn execution of another using the "fork" instruction).

8. Borkenhagen and Torii are analogous art because they are both in the same field of endeavor of microprocessor systems.

9. It would have been obvious to a person having ordinary skill in the art at the time of invention having the teachings of Borkenhagen and Torii before him or her to employ the "fork" instruction that spawns execution of a new thread from within a thread as taught by Torii in the system of Borkenhagen. The motivation for doing so would have been to implement the task to be performed in a multi-threaded fashion, so if one of the sub-tasks encounters a long-latency event, one of the other sub-tasks can execute.

10. As to claim 11, Borkenhagen discloses a processing block, an execution method, comprising:

executing the first instruction, and as part of the execution of the first instruction, adding a first thread entry in a thread array of a thread switching structure, if the first instruction is a create thread instruction spawning interleaved execution of a second thread of instructions with execution of other threads (col. 24, claim 1, before the first thread starts executing, the state of the thread is stored in at least one hardware register, i.e., the thread state register. The state of the thread will be stored regardless of how the thread is created, so the thread array of Borkenhagen's system will include the parent thread and the child threads, thus, it has the required functionality), the thread switching structure being disposed and maintained within the processing sub-block (col. 13, lines 33-67, col. 14, lines 1-2, the thread structure is the collection of bits in the thread state register, col. 6, lines 24-28, the state of the thread in the thread stage register will be

updated when a long latency event occurs for that thread, so the contents of the thread stage register are maintained) to facilitate interleaved execution of threads of instructions by the processing sub-block, and the first thread entry describing the second thread (Figure 4A, #400, the state registers comprise the thread array and each register is considered to be a thread entry in the thread array. Furthermore, each state register describes the corresponding thread. In the system of Borkenhagen the first thread entry will be for the parent thread and not for the child thread, the second thread entry will describe the child thread, i.e., the second thread. Nonetheless, the information for each child thread will be contained in the thread array of Borkenhagen).

11. Borkenhagen does not disclose a create thread instruction that spawns execution of a second thread from within a thread, nor fetching a first instruction of a first thread of instructions However, Torii discloses a create thread instructions (col. 4, lines 65-67, child threads can spawn execution of another using the "fork" instruction).

12. It would have been obvious to a person having ordinary skill in the art at the time of invention having the teachings of Borkenhagen and Torii before him or her to employ the "fork" instruction that spawns execution of a new thread from within a thread as taught by Torii in the system of Borkenhagen. The motivation for doing so would have been to implement the task to be performed in a multi-threaded fashion, so if one of the sub-tasks encounters a long-latency event, one of the other sub-tasks can execute.

13. Although Borkenhagen does not explicitly disclose an execution method that

includes fetching a first instruction of a first thread of instructions, a first thread of instructions must have been fetched from external memory and loaded into the processor memory in order for a thread to execute. It would have been obvious to one having ordinary skill at the time of invention that a first instruction of a first thread of instructions must be fetched in order for the first instruction of thread to execute.

14. As to claim 6, Borkenhagen discloses that the processing block further comprises an interface to couple the processing block to an external set of registers (Figure 2, the execution unit, here the fixed point unit, is connected to the GPRs and SPRs, the processing block is consider to be the aggregate of the thread switch logic, the transition cache, the L1 D-cache, the L2 cache, main memory, the L-1 instruction cache, and the instruction unit).

15. As to claim 10, Borkenhagen discloses that the processing sub-block further comprises another storage sub-block coupled to the execution sub-block, to store instructions of the threads (Figure 2, #150, L-1 Instruction cache).

16. As to claims 2 and 12, Borkenhagen does not disclose that each thread entry comprises a thread program counter to identify an instruction of the corresponding described thread as a current instruction to be executed, when the corresponding described thread is being executed. However, it would have been obvious to a person having ordinary skill in the art at the time of invention that a thread program counter is implicit in the system of Borkenhagen. The system has to know which instruction of a given thread to execute when a thread resumes operation.

17. Claims 3-4, 13, and 15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Borkenhagen and Torii as applied to claims 1 and 11 above, in view of Proskauer et al. (US 5,673,272), hereinafter Proskauer.

18. As to claims 3 and 13, neither Borkenhagen nor Torii disclose that each thread entry comprises an activeness indicator indicating whether the corresponding described thread is currently in an active state or an inactive state, where the corresponding described thread is to be included among the threads to be interleavingly executed by the execution sub-block, while the thread is in the active state, and not included, while the thread is in the inactive state. However, Proskauer discloses that each thread entry comprises an activeness indicator indicating whether the corresponding described thread is currently in an active state or an inactive state, where the corresponding described thread is to be included among the threads to be interleavingly executed by the execution sub-block, while the thread is in the active state, and not included, while the thread is in the inactive state (col. 9, lines 55-67, once the thread is created it is in the active state and when it finishes execution it enters the inactive state)

19. Borkenhagen, Torii and Proskauer are analogous art because they are in the same field of endeavor of microprocessor systems.

20. It would have been obvious to a person having ordinary skill in the art at the time of invention having the teachings of Borkenhagen, Torii and the teachings of Proskauer before him or her to incorporate the activeness indicator of Proskauer in the system of

Borkenhagen and Torii. The motivation for doing so would have been to be able to reuse thread data from the thread entry fields when a thread goes from inactive to active (this will happen when a 2nd capture occurs, see Proskauer, col. 9, lines 3-7). In the system of Proskauer, the thread entries contain thread-specific global data, a thread active flag, a data pointer, and a buffer for passing data between a plurality of active threads (col. 16, claim 7). Each thread entry corresponds to one of the capture instruments.

21. As to claims 4 and 15, neither Borkenhagen nor Torii disclose that the thread management sub-block is equipped to reset the activeness indicator of a thread from the active state to the inactive state as part of the execution of a thread termination instruction of a thread terminating its own execution. However, Proskauer discloses that the thread management sub-block is equipped to reset the activeness indicator of a thread from the active state to the inactive state as part of the execution of a thread termination instruction of a thread terminating its own execution (col. 10, lines 6-10, the side thread becomes inactive when it finishes executes its test functions, and then it is attached to the free queue, and a thread terminating its own execution is equivalent to the thread finishing execution, in either case the thread stops executing instructions)

22. It would have been obvious to a person having ordinary skill in the art at the time of invention having the teachings of Borkenhagen, Torii, and the teachings of Proskauer before him or her to modify the system of Borkenhagen and Torii to incorporate the resetting of the activeness indicator of Proskauer, allowing the activeness indicator to

be reset to inactive when a thread terminates its own execution. Otherwise the thread would never be freed.

23. Claims 5 and 14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Borkenhagen in view of Torii as applied to claims 1 and 11 above, and further in view of Akkary et al. (WO 99/31580), hereinafter Akkary.

24. As to claims 5 and 14, neither Borkenhagen nor Torii disclose that each thread entry comprises thread dependency information describing at least a plurality of registers of an external set of registers, on which the corresponding described thread depends. However, Akkary discloses that each thread entry comprises thread dependency information describing at least a plurality of registers of an external set of registers, on which the corresponding described thread depends (p. 13, 2nd paragraph, lines 3-6, the thread management logic eventually needs to determine the true program order. p. 14, 4th paragraph, lines 3-8, an instruction queue array receives instructions from a particular thread, and instructions that are part of another thread are written into an instruction queue in another trace buffer, the instruction queue array is shown in Fig. 12 and includes an instruction ID field. There is also a data and dependency array (DAD array) that includes entries with instruction ID's that have a one-on-one correspondence with the instruction ID's in the instruction queue array, see p. 16, 5th paragraph, lines 1-3. The DAD array has dependency fields for each of the logical registers, see Fig. 13, and is used by the thread management logic to ensure that the threads eventually execute in the correct order to guarantee data consistency).

25. Borkenhagen, Torii, and Akkary are analogous art because they are in the same field of endeavor of multiprocessor systems.

26. It would have been obvious to a person having ordinary skill in the art at the time of invention having the teachings of Borkenhagen, Torii, and the teachings of Akkary before him or her to incorporate the thread dependency method of Akkary in the system of Borkenhagen and Torii in order to allow out-of-order execution, thereby increasing the performance of the processor (Akkary, Background of Invention, 2nd paragraph).

27. Claims 7 and 16 are rejected under 35 U.S.C. 103(a) as being unpatentable over Borkenhagen in view of Torii as applied to claims 1 and 11 above, and further in view of Parady (US 5,933,627).

28. As to claims 7 and 16, Borkenhagen discloses that the thread management sub-block is further equipped to select a non-current one of the plurality of threads to be the new current thread to be executed, updating the current thread identifier and switching execution to a first instruction of the new current thread accordingly (col. 6, lines 14-19, the states of all threads are stored and part of the state is whether the thread is an active thread or a background thread waiting for execution).

29. Neither Borkenhagen nor Torii disclose a thread switching instruction of a thread that instructs the execution sub-block to switch execution to another thread. However, Parady discloses a thread switching instruction of a thread that instructs the execution

sub-block to switch execution to another thread (col. 6, lines 47-53, designated instruction types have a program address register identified in a thread switch instruction field, and these program address registers hold the addresses of threads. The switch logic will switch execution to the thread in the program address register, if a switch signal that is generated in response to the designated instructions is asserted. Here the switch signal will be asserted on a level two cache miss).

30. Borkenhagen, Torii, and Parady are analogous art because they are in the same field of endeavor of microprocessor systems.

31. It would have been obvious to a person having ordinary skill in the art at the time of invention having the teachings of Borkenhagen, Torii and Parady before him or her to incorporate the thread switch instruction field concept of Parady into the system of Borkenhagen and Torii to allow a thread to specify which thread should execute next when a long latency event occurs (Parady, col.2, lines 25-29).

32. Claims 8 and 17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Borkenhagen in view of Torii and Parady as applied to claims 7 and 16 above, and further in view of Penkovski et al. (US 7,089,340), hereinafter Penkovski.

33. As to claims 8 and 17, Parady discloses that the execution sub-block is equipped to select the next current thread on a selected one of a round-robin basis (col. 2, lines 30-34, in the alternate embodiment, the program address registers are switched in a round-robin fashion).

34. Neither Borkenhagen, Torii, nor Parody disclose that the execution sub-block is equipped to select the next current thread on a basis of fixed priority basis and a rotating priority basis. However, Penkovski discloses that the next thread to execute can be based on chosen priority rules (col. 3, lines 59-63).

35. Borkenhagen, Torii, Parady, and Penkovski are analogous art because they are all in the same field of endeavor of microprocessor systems.

36. It would have been obvious to a person having ordinary skill in the art at the time of invention having the teachings of Borkenhagen, Torii, Parady, and the teachings of Penkovski before him or her to incorporate priority-based scheduling systems in the combined system of Borkenhagen, Torii, and Parady. The motivation for doing so would have been to provide more flexibility in the scheduling of threads.

37. Claim 9 is rejected under 35 U.S.C. 103(a) as being unpatentable over Borkenhagen in view of Torii as applied to claim 1 above, and further in view of Andrews et al. (US 6,233,643), hereinafter Andrews.

38. As to claim 9, neither Borkenhagen nor Torii disclose that processing block further comprises an input/output interface configurable to be a selected one of an input interface and an output interface to particularize the processing block as a selected one of an input processing block and an output processing block of a signal processing macroblock. However, Andrews discloses that the processing block further comprises an input/output interface configurable to be a selected one of an input interface and an

output interface to particularize the processing block as a selected one of an input processing block and an output processing block of a signal processing macroblock (col. 8, lines 16-25, the sixteen bits of the GPIO port 70 of each DSP can be programmed to be input only or output only, the bits are connected to a RAM buffer, the DSPs can read or write the RAM buffer. Col. 8, lines 45-48, there is also another GPIO port that is used as a data bus and control bus to read and write to registers in the network framer. Thus, these ports to the DSPs are configurable to be input or output)

39. Borkenhagen, Torii, and Andrews are analogous art because they are both in the same field of endeavor of microprocessors systems.

40. It would have been obvious to a person having ordinary skill in the art at the time of invention having the teachings of Borkenhagen, Torii, and the teachings of Andrews before him or her to incorporate the configurable interface (GPIO port) of Andrews in the system of Borkenhagen and Torii in order to allow the microprocessor to communicate with external devices.

41. Claims 18-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rodriguez (US 7,136,417) in view of Borkenhagen.

42. As to claim 18, Rodriguez discloses a signal processing macroblock comprising a set of registers, and at least a selected one of an input processing block coupled to the set of registers and an output processing block coupled to the set of registers (Fig. 2, the compression engine #217 is the input processing block and the decompression

engine is the output processing block and both are both coupled to the memory #249 through the common bus. Col. 14, lines 65-67, the compression engine receives data from the video decoder, col. 16, lines 46-49, the decompression engine receives data from the compression engine. Thus the decompression engine is considered to be the output processing block).

43. Rodriguez does not necessisarily disclose that the input and output processing blocks include execution and thread management facilities equipped to support interleaved execution of multiple threads of instructions. On the other hand, Borkenhagen discloses a processing block that includes execution and thread management facilities equipped to support interleaved execution of multiple threads of instructions.

44. Rodriguez and Borkenhagen are analogous art because they are both in the field of endeavor of microprocessor systems.

45. It would have been obvious to a person having ordinary skill in the art at the time of invention having the teachings of Rodriguez and Borkenhagen before him or her to incorporate the processing block that includes execution and thread management facilities equipped to support interleaved execution of multiple threads of instructions as taught by Borkenhagen in signal processing macroblock of Rodriguez in order to avoid stalling the processor when long-latency events occur (Borkenhagen, col. 6, lines 23-26).

46. As to claim 19, this claim is rejected for the same reasons as claim 18 above. In addition, Rodriguez discloses that the signal processing macroblock further comprises a computation block coupled to the set of registers (Fig. 2, the signal processing block #214 is a computation block) and Borkenhagen discloses a computation block including execution and thread management facilities equipped to support interleaved execution of multiple threads of instructions, including instructions performing mathematical operations (Borkenhagen, Fig. 3, col. 11, line 4, the execution unit #280 is a floating point unit that performs mathematical operations, and Fig. 3, #400, is the thread management facility, and both the execution unit and the thread logic support interleaved execution of multiple threads of instructions)

47. As to claim 20, Rodriguez does not discloses that at least a selected one of the facilities of the input processing block, the output processing block and the computation block, equipped to support interleaved execution of multiple threads, includes a storage sub-block to store a thread switching structure that includes a current thread identifier identifying one of the multiple threads as a current thread to be executed, and a thread array including thread entries describing corresponding ones of the multiple threads .

48. However, Borkenhagen discloses that at least a selected one of the facilities of the input processing block, the output processing block and the computation block, equipped to support interleaved execution of multiple threads, includes a storage sub-block to store a thread switching structure (col. 13, lines 33-67, col. 14, lines 1-2,

the thread structure is the collection of bits in the thread state register) that includes a current thread identifier identifying one of the multiple threads as a current thread to be executed (col. 6, lines 14-19), and a thread array including thread entries describing corresponding ones of the multiple threads (Figure 4A, #400, the state registers comprise the thread array and there is one per thread).

49. It would have been obvious to a person having ordinary skill in the art at the time of invention having the teachings of Rodriguez and Borkenhagen before him to incorporate a storage sub-block to store a thread switching structure that includes a current thread identifier identifying one of the multiple threads as a current thread to be executed,and a thread array including thread entries describing corresponding ones of the multiple threads. The motivation for doing so would be the same as that of claim 18, since the storage sub-block is needed to implement the multi-threaded system of Borkenhagen.

50. As to claim 21, Rodriguez does not disclose that at least a selected one of the facilities of the input processing block, the output processing block and the computation block, equipped to support interleaved execution of multiple threads, further includes an execution sub-block equipped to create a thread entry in the thread array for a thread as part of the execution of a create thread instruction spawning interleaved execution of the thread,

51. However, Borkenhagen discloses that at least a selected one of the facilities of the input processing block, the output processing block and the computation

block, equipped to support interleaved execution of multiple threads, further includes an execution sub-block equipped to create a thread entry in the thread array for a thread as part of the execution of a create thread instruction spawning interleaved execution of the thread (col. 12, lines 41-45, Figure 4A and 4B, the instruction unit and the storage control unit are connected to the thread switch logic and data bits from these units update the thread state registers with the new state of a spawned thread, i.e., create thread entries and add them to the thread array. The create thread instruction is implicit in the existence of a new thread, i.e., if a new thread has been created, an instruction must have created it).

52. It would have been obvious to a person having ordinary skill in the art at the time of invention having the teachings of Rodriguez and Borkenhagen before him to incorporate the execution sub-block equipped to create a thread entry in the thread array for a thread as part of the execution of a create thread instruction spawning interleaved execution of the thread as taught by Borkenhagen in the system of Rodriguez in order to implement the multi-threaded system of Borkenhagen. The motivation for storing a thread entry in the thread array is to provide information about the state of each thread (Borkenhagen, Fig 4A, #442, #444, are the thread entries that store the state of the threads).

53. Claim 22 is rejected under 35 U.S.C. 103(a) as being unpatentable over Rodriguez in view of Borkenhagen as applied to claim 20 above, and further in view of Proskauer.

54. As to claim 22, neither Rodriguez nor Borkenhagen disclose that at least a selected one of the facilities of the input processing block, the output processing block and the computation block, equipped to support interleaved execution of multiple threads, further includes an execution sub-block equipped to reset an activeness indicator of a thread entry in the thread array for a thread from indicating an active state to indicating an inactive state as part of the execution of a thread termination instruction terminating execution of the thread.

55. However, Proskauer discloses that at least a selected one of the facilities of the input processing block, the output processing block and the computation block, equipped to support interleaved execution of multiple threads, further includes an execution sub-block equipped to reset an activeness indicator of a thread entry in the thread array for a thread from indicating an active state to indicating an inactive state as part of the execution of a thread termination instruction terminating execution of the thread (col. 10, lines 6-10, the side thread becomes inactive when it finishes executing its test functions, and then it is attached to the free queue, and a thread terminating its own execution is equivalent to the thread finishing execution, in either case the thread stops executing instructions).

56. Rodriguez, Borkenhagen, and Proskauer are all analogous art because they are in the same field of endeavor of microprocessor systems.

57. It would have been obvious to a person having ordinary skill in the art at the time

of invention having the teachings of Rodriguez and Borkenhagen, and the teachings of Proskauer before him or her to incorporate the resetting of the activeness indicator of Proskauer, allowing the activeness indicator to be reset to inactive when a thread terminates its own execution. The motivation for doing so would have been to avoid threads never being freed even though they have finished executing.

58. Claim 23 is rejected under 35 U.S.C 103(a) as being unpatentable over Rodriguez in view of Borkenhagen as applied to claim 20 above, and further in view of Parady.

59. As to claim 23, neither Rodriguez nor Borkenhagen disclose that at least a selected one of the facilities of the input processing block, the output processing block and the computation block, equipped to support interleaved execution of multiple threads, further includes an execution sub-block equipped to select a thread as a new current thread to be executed, updating a current thread identifier of the thread switching structure to identify the selected thread, and switching to execute an instruction of the selected thread, as part of the execution of a thread execution switching instruction.

60. However, Borkenhagen discloses that the execution sub-block is equipped to select a thread as a new current thread to be executed, updating a current thread identifier of the thread switching structure to identify the selected thread (col. 6, lines 14-19, the states of all threads are stored and part of the state is whether the thread is

an active thread or a background thread waiting for execution. Col. 6, lines 17-21, the system is able to change the state of the active thread, and this includes switching execution to one of the background threads), but Borkenhagen does not disclose that selecting a new thread results as part of a thread switching instruction.

61. Neither Rodriguez nor Borkenhagen disclose a thread execution switching instruction. However, Parady discloses a thread execution switching instruction (col. 6, lines 47-53, designated instruction types have a program address register identified in a thread switch instruction field, and these program address registers hold the addresses of threads. The switch logic will switch execution to the thread in the program address register, if a switch signal that is generated in response to the designated instructions is asserted. Here the switch signal will be asserted on a level two cache miss).

62. Rodriguez, Borkenhagen, and Parady are all analogous art because they are all in the same field of endeavor of microprocessor systems.

63. It would have been obvious to a person having ordinary skill in the art at the time of invention having the teachings of Rodriguez, Borkenhagen and Parady before him or her to combine the system of Rodriguez and Borkenhagen to employ the switching instruction of Parady to allow the threads to communicate with the thread management logic to indicate that a thread switch should occur due to a possible long-latency event that may occur during the execution of the last instruction executed in the thread (Parady, col. 2, lines 25-29, a switching mechanism to switch in response to long-latency events, col. 6, lines 47-53, designated instructions have a thread switch

instruction field that they can use to specify which program address register to use if the designated instruction causes a long-latency event).

64. Claims 24-27, and 30-33, and 36 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rodriguez in view of Borkenhagen and further in view of Davis et al. (US 4,991,169), hereinafter Davis.

65. As to claim 24, Rodriguez discloses a media processor comprising a plurality of signal processing units comprising a set of registers, an input processing block coupled to the set of registers, including an input interface, an execution facility, and an output processing block coupled to the set of registers, including an output interface and an execution facility (Abstract, lines 1-2, the system processes a video signal, i.e., a media. Fig. 2, the compression engine #217 is the input processing block and the decompression engine is the output processing block and both are coupled to the memory #249, i.e., a set of registers, through the common bus. col. 14, lines 65-67, the compression engine receives data from the video decoder, col. 16, lines 46-49, the decompression engine receives data from the compression engine. Thus, the decompression engine is considered to be the output processing block).

66. Rodriguez does not disclose execution and thread management facilities equipped to support interleaved execution of multiple threads of instructions. However, Borkenhagen discloses execution and thread management facilities equipped to support interleaved execution of multiple threads of instructions (Abstract, lines 1-3).

67. Rodriguez, and Borkenhagen are analogous art because they are both in the same field of endeavor of microprocessor systems.

68. It would have been obvious to a person having ordinary skill in the art at the time of invention having the teachings of Rodriguez and Borkenhagen before him or her to incorporate the multithreaded system of Borkenhagen with the system of Rodriguez. The motivation for doing so would have been to improve performance of the processing by keeping long-latency event from blocking execution of other instructions (Borkenhagen, col. 6, lines 23-26).

69. Neither Rodriguez nor Borkenhagen discloses a plurality of signal processing units coupled to the direct memory access unit. However, Davis discloses a plurality of signal processing units coupled to the direct memory access unit (Fig. 14A, #220, DMA address indicates a direct memory access unit coupled to DSP1).

70. It would have been obvious to a person having ordinary skill in the art at the time of invention having the teachings of Rodriguez, and Borkenhagen, and the teachings of Davis before him or her to incorporate the direct memory address (DMA) unit of Davis in the system of Rodriguez and Borkenhagen in order to allow for fast memory transfers between the memory and the signal processing units. It is well known in the art that DMA units provide fast data transfers.

71. As to claim 30, Rodriguez discloses a a system comprising: a host processor; first memory coupled to the host processor; second memory; a media processor

coupled to the second memory and the host processor (Fig. 2, #244 is the host processor, the memory #252 is coupled to the host processor, the second memory #251 is coupled to the host processor #244 and the media processor (comprised of all components below the bus #205, i.e., the compression engine, the signal processing system, the decompression engine, etc.), the media processor having a plurality of signal processing units, at least a first of which signal processing units comprises a set of registers, including an input interface, and an output processing block coupled to the set of registers, including an output interface (Fig. 2, the input processing block is the compression engine #217, it receives data from the video decoder, see col. 14, lines 65-67. The output processing block is the decompression engine #222, since it receives data from the compression engine, see col. 16, lines 46-49, and the output processing block is coupled to a set of registers #273 (the storage devices comprise a set of registers). There are a plurality of signal processing units, the compression engine together with the decompression engine is considered to be the first one, and the signal processing system #214 is considered to be a second one. There is an input interface between the analog video decoder and the compression engine and there is inherently an output interface between the decompression engine and the bus #205).

72. Rodriguez does not disclose execution and thread management facilities equipped to support interleaved execution of multiple threads of instructions. However, Borkenhagen discloses execution and thread management facilities equipped to

support interleaved execution of multiple threads of instructions (Abstract, lines 1-3).

73. It would have been obvious to a person having ordinary skill in the art at the time of invention having the teachings of Rodriguez and Borkenhagen before him to incorporate the execution and thread management facilites to support multithreading. The motivation for doing so would have been to The motivation for doing so would have been to improve performance of the processing by keeping long-latency event from blocking execution of other instructions (Borkenhagen, col. 6, lines 23-26).

74. Neither Rodriguez nor Borkenhagen disclose the media processor having at least a direct memory access unit to access media data, and a plurality of signal processing units coupled to the direct memory access unit to process the accessed media. However, Davis discloses having at least a direct memory access unit to access data from memory, and a plurality of signal processing units coupled to the direct memory access unit to process the data (Fig. 14A, #220, DMA address indicates a direct memory access unit coupled to DSP1).

75. It would have been obvious to a person having ordinary skill in the art at the time of invention having the teachings of Rodriguez, and Borkenhagen, and the teaching of Davis before him or her to incorporate the direct memory address (DMA) unit of Davis in the system of Rodriguez and Borkenhagen in order to allow for fast memory transfers between the memory and the signal processing units. It is well known in the art that DMA units provide fast data transfers.

76. As to claims 25 and 31, the claims are rejected for the same reasons as claims 24 and 30 above. In addition, Rodriguez discloses that the first signal processing unit further comprises a computation block coupled to the set of registers, including execution and thread management facilitates equipped to support interleaved execution of multiple threads of instructions, including instructions performing mathematical operations (Fig. 2, #214, the signal processing system is a computation block that performs mathematical operations, and Borkenhagen discloses multithreaded processors with thread management facilities equipped to support interleaved execution of multiple threads of instructions (Abstract, lines 1-3), and the system processing system comprises a processor.

77. As to claims 26 and 32, the claims are rejected for the same reasons as claims 25 and 31 above. In addition, Borkenhagen discloses a processing block equipped to support interleaved execution of multiple threads, includes a storage sub-block to store a thread switching structure that includes a current thread identifier identifying one of the multiple threads as a current thread to be executed, and a thread array including thread entries describing corresponding ones of the multiple threads (Abstract, lines 1-3, multithreaded processor with thread switching logic, the storage sub-block is comprised of the thread state registers . Col. 13, lines 33-67, col. 14, lines 1-2, the thread structure is the collection of bits in the thread state register. Col. 6, lines 9-14, the state of the thread includes an indication of whether the thread is an active thread or a background thread waiting to be executed. The thread stage registers comprise the thread array).

78. As to claims 27 and 33, the claims are rejected for the same reasons as claims 25 and 31 above. In addition, Borkenhagen discloses a processing block that includes an execution sub-block equipped to create a thread entry in the thread array for a thread as part of the execution of a create thread instruction spawning interleaved execution of the thread (Figures 4A and 4B shows that the inputs into the thread state registers, which are the thread entries, are output from the instruction unit and the storage control unit. The instruction and storage control units comprise an execution sub–block, and the thread state register does not become valid until a new thread is created. It is not disclosed how threads are created, but they could be created by the well-known "fork" instruction that is implemented in multithreading operating systems, which spawns new threads from within a thread. Regardless of how the thread is created, a thread state register will be updated before the thread starts executing. Although the system of Borkenhagen has entries for the parent and the child threads (unlike what is described in the claim), Borkenhagen's thread array satisfies in that a thread entry will be created in the thread array when a thread is spawned from within another thread).

79. As to claim 36, Rodriguez discloses that the system is a selected one of a server, a palm sized personal digital assistant, a wireless mobile phone, a set-top box, an entertainment control console, a video recorder, or a video player (Fig. 2, #200, col. 1, lines 18-28, the system of Rodriguez is a DHCT, otherwise known as a set-top box).

80. Claims 28 and 34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rodriguez in view of Borkenhagen, and Davis as applied to claims 25 and 31 above, and further in view of Proskauer.

81. As to claims 28 and 34, neither Rodriguez, Borkenhagen nor Davis disclose that at least a selected one of the facilities of the input processing block, the output processing block and the computation block, equipped to support interleaved execution of multiple threads, further includes an execution sub-block equipped to reset an activeness indicator of a thread entry in the thread array for a thread from indicating an active state to indicating an inactive state as part of the execution of a thread termination instruction terminating execution of the thread.

82. However, Proskauer discloses an execution sub-block equipped to reset an activeness indicator of a thread entry in the thread array for a thread from indicating an active state to indicating an inactive state as part of the execution of a thread termination instruction terminating execution of the thread (col. 9, lines 55-67, once the thread is created it is in the active state and when it finishes execution it enters the inactive state).

83. Rodriguez, Borkenhagen, Davis, and Proskauer are all analogous art because they are all in the same field of endeavor of microprocessor systems.

84. It would have been obvious to a person having ordinary skill in the art at the time

of invention having the teachings of Rodriguez, Borkenhagen, and Davis, and the teachings of Proskauer before him to incorporate the activeness indicator of Proskauer in the system of Rodriguez, Borkenhagen and Davis. The motivation for doing so would have been to facilitate reusing thread data from the thread entry fields when a thread goes from inactive to active (this will happen when a 2nd capture occurs, see Proskauer, col. 9, lines 3-7). In the system of Proskauer, the thread entries contain thread-specific global data, a thread active flag, a data pointer, and a buffer for passing data between a plurality of active threads (Proskauer, col. 16, claim 7). Each thread entry corresponds to one of the capture instruments.

85. Claims 29 and 35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rodriguez in view of Borkenhagen, and Davis as applied to claims 25 and 31 above, and further in view of Parady.

86. As to claims 29 and 35, the claims are rejected for the same reasons as claims 25 and 31 above. In addition, Borkenhagen discloses an execution sub-block equipped to select a thread as a new current thread to be executed, updating a current thread identifier of the thread switching structure to identify the selected thread, and switching to execute an instruction of the selected thread (col. 6, lines 14-19, the states of all threads are stored and part of the state is whether the thread is an active thread or a background thread waiting for execution).

87. Neither Rodriguez, Borkenhagen, nor Davis disclose a thread execution switching instruction that allows the current thread to signal the execution sub-block to

select a new current thread to be executed. However, Parady discloses a thread execution switching instruction that allows the current thread to signal the execution sub-block to select a new current thread to be executed (Parady, col. 6, lines 47-53, designated instructions have a thread switch instruction field)

88. It would have been obvious to a person having ordinary skill in the art at the time of invention having the teachings of Rodriguez, Davis, Borkenhagen, and the teachings of Parady before him or her to incorporate the execution switching instruction of Parady in the system of Rodriguez, Davis, and Borkenhagen in order to allow the threads to communicate with the thread management logic to indicate that a thread switch should occur due to a possible long-latency event that may occur during the execution of the last instruction executed in the thread (Parady, col. 2, lines 25-29, a switching mechanism to switch in response to long-latency events, col. 6, lines 47-53, designated instructions have a thread switch instruction field that they can use to specify which program address register to use if the designated instruction causes a long-latency event).

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David A. Holloway whose telephone number is (571)270-1899. The examiner can normally be reached on mon-fri 8:00 am - 5:00 pm (alternate fridays off).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Nabil El-Hady can be reached on (571)272-3963. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

DH 06/12/2007

N. El-Hady

NABIL M. EL-HADY
SUPERVISORY PATENT EXAMINER